CSCI 5832

Natural Language Processing

Assignment 3: Named Entity Recognition

Fall 2021

HW 3 Group 16

Nitin Kumar
Rohan Das
Tushar Gautam

Model 1 - DeepCRF - BiLSTM-CNN-CRF

Introduction

We started off with the DeepCRF approach to have a baseline while we continued to work on other transformer based (pre trained) models which are known to be more effective for the given task.

In DeepCRF approach, The model consists of a bi directional LSTM (which has access to context from both past and future) with convolutional neural network (for extracting character-level representations of words) and CRF to consider correlations among the nearby token. It optionally also takes word embedding eg. Glove for embedding the input words. We used the DeepCRF model starting with words as input features and then improved it by using word embeddings (Glove) in the model.

Approach

1. For the pre-processing step, the dataset was shuffled and divided into train/dev/test dataset in 80/10/10 proportion.
2. We considered two approaches for input word representation ie. a.) to create a representation using vocabulary with notations for unknown words and also the consideration of padding. b.) to use glove word embedding to represent the input words.
3. We trained the model using 80 % (train) and 10 % dev data for both the cases a and b defined above.

4. For case a.) when input words were represented using vocabulary: We trained the model using Adam method with batch size = 32, learning rate = 0.001 and total iterations of 50 with number of layers = 1 and hidden nodes = 200.
5. For case b.) when Glove word embedding was used: We trained the model using both Adam and SGD optimizer and SGD gave superior results for training (F1 measure) but was overfitting on the data as the test F1 measure was low, whereas Adam gave better results for the test data. We trained for 50 iterations with batch size of 32, learning rate = 0.001, number of layers = 1 and hidden nodes = 200 and Glove word embedding length of 100. We used f1 accuracy during training to save the superior model.
6. We tested using the shuffled 10% test data and computed the f1 accuracy using the script provided.

Results

1. For Case a.) DeepCRF with words representation using vocabulary: For the held out test data, the following are the observations from the script provided for evaluation:

```
1808  entities in gold standard.
1824  total entities found.
1361  of which were correct.

Precision:  0.7461622807017544
Recall:  0.7527654867256637
F1-measure:  0.7494493392070484
```

2. For Case b.) DeepCRF with Glove word embeddings: For the held out test data, the following are the observations:

```
808  entities in gold standard.
1824  total entities found.
1425  of which were correct.

Precision:  0.78125
Recall:  0.7881637168141593
F1-measure:  0.7846916299559471
```

Discussion

1. For Case a.) DeepCRF with word representation using vocabulary: Model's accuracy was affected due to Unknown words. Also, the word representation using vocabulary was inhibiting the overall accuracy of the model. So, We moved to glove word embeddings which improved the accuracy of the model significantly.

2. For Case b.) With SGD, the model was performing really well during the training but the test performance was not good ie. much below the training performance. The model was overfitting. We used Adam optimizer which helped to generalize the model better and thus provided better results for the test as shown above.

Model 2 - Fine-tuning on BERT base model (Huggingface)

Introduction

For named entity recognition task, transformer models like Bert have been the state of the art. Our idea was to leverage the pre-trained Bert model and fine-tune with a given medical data for the assignment. We fine-tuned the case-sensitive Bert model which is pre-trained on a large corpus of Google Book Corpus and English Wikipedia. We used the model from HuggingFace to perform the fine-tuning. This experiment further laid the foundation for exploring BioBERT (Model 3) which has been pre-trained on corpora from the biomedical domain and hence expected to perform better on the dataset given to us.

Approach

1. Before we could begin training, as a preprocessing step, first the dataset was shuffled and split with 80/10/10 % to form the train/dev/test data set.
2. Furthermore, the splits were tokenized with Bert tokenizer and respective labels extrapolated to align with each subword. Bert employs subword (Wordpiece) tokenization which splits a word into its constituent subwords viz. “SATB1” is split into two words “SAT” “##B” and “##1". To maintain the word label, we assigned the “B" tag to the first subword and “I" tag to the rest of the subwords. So if “SATB1” was assigned “B” tag, “SAT” will be assigned “B” and “##B” and “##I” tokens will be assigned “I” tag. The tokenizer adds “##” as a prefix to mark the subwords.
3. Besides tokenization, we also considered padding and truncation to ensure token size is within Bert max token limit. Furthermore, dynamic padding was added to ensure tokens are padded only with respect to the batch data during training.
4. Finally, the pre-trained model was fine-tuned for 4 epochs and train batch size of 8.
5. Model was published on Hugging Face and has further details on the hyperparameters and quick inference feature. HuggingFace Repository.

6. Code available under *bert-base/bert_base_finetune_ner.ipynb* notebook.

Results

1. The tag level precision/recall/F1 for the model was 0.79/0.83/0.80 on validation dataset and 0.78/0.82/0.80 on hold-out test dataset.
2. The entity level precision/recall/F1 for the model was 0.08/0.08/0.08 which is far worse than what we observed for tag level. Possible reasons with exploratory measures explained in the "challenges" section below.

Discussion

Particularly for this experiment, we faced an issue with the way Bert tokenization treated special characters which is different from the assignment dataset. Problems with fixes have been outlined below:

1. The Bert tokenizer splits subsequent special characters such as "%.", ".+" as separate tokens but the dataset given for the assignment didn't have this split. As the assignment evaluation script considers entity level metrics, even a minor change in entity position would throw-off the F1 score for the entire set.
2. Another issue was that such special character split inconsistencies caused all subsequent sentences' entity position to get computed inaccurately. This further exacerbated the metric.
3. To fix #1 / #2 required changing evalNER.py which has been shared in Piazza post. As updating evalNER.py wasn't the goal of the assignment, we soon enough dropped this approach in favor of an alternate option as described below.
4. As an alternate approach, we merged the subsequent special characters in the prediction and assigned tag final as that of the first special character. This took care of issues from #1 / #2 for the most part and increased F1 from 0.01 to 0.08.
5. Rest of the challenge was debugging why we were seeing such a high F1 score for tag level but low F1 score for entity level. Looking carefully at some of the predictions we noticed minor inaccuracies at tag level in a sentence would have significant impact at the entity level. So suppose if a token was predicted as 'O' but it has been originally tagged as 'I', while the rest of the tokens in a sentence have been tagged accurately it would mean 0 precision/recall for entity level metrics. Spending considerable time it seemed the model required further fine-tuning with biomedical data or rather choose a pre-trained model which was trained on biomedical data. Bio-BERT is one such model we explored in Model 3 below.

Model 3 - Fine-tuning on BioBERT

Introduction

In this approach we use BioBERT (specifically v1.1), which is a pre-trained language representation model for the biomedical domain. Essentially BioBERT was first initialized with weights from BERT, which was pre-trained on general domain corpora (English Wikipedia and BooksCorpus). Then, BioBERT was pre-trained on biomedical domain corpora (PubMed abstracts and PMC full-text articles). We use this pre-trained model and fine-tune it on a train-test-dev split of the data that we are given and evaluate it on the Named Entity Recognition task.

Approach

1. We used the same train-test-dev splits (80/10/10) that we used for Model 1. Additionally, we generated a file with the training and development sets concatenated as one of the requirements to fine-tune this model.
2. As seen with Model 2, BioBERT also employs a form of subword modeling, called WordPiece tokenization, to deal with unknown words. We modified the tokenization algorithm for this model to ensure that consecutive punctuation (within the same token or in consecutive tokens) are left unaltered as this can affect proper evaluation on the given evaluation script.
3. We then fine-tuned BioBERT on our training set, and evaluated over the development set.
4. Next, we ran inference on the fine-tuned model for the unseen test set. This produces two files as output: the predicted tags and the corresponding tokens (as per the tokenization scheme employed by the model).
5. We passed the two output files obtained above through the de-tokenization script (./biocodes/ner_detokenize.py) provided to us by the makers of this model, to obtain the word level prediction file.
6. We then evaluated the output of the previous step on the entity level evaluation script provided by the makers of the model, and compared it with the benchmarks over several popular NER related Biomedical corpus.
7. Finally, we do some post processing on the predicted output file to remove the duplicate tag column and evaluate against the gold tags using the evaluation script provided by this assignment.

Results

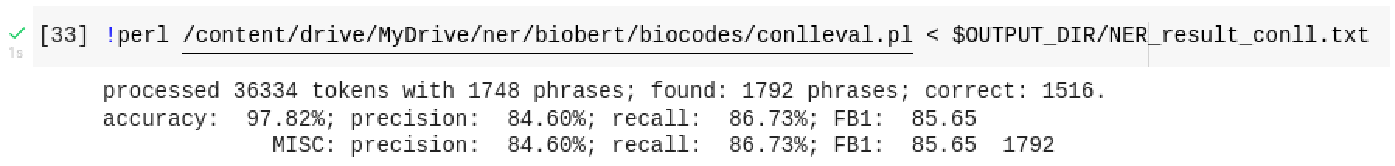
- Token level evaluation results on the development set were as follows:

  F1 = 0.9073543, Precision = 0.89881563, Recall = 0.91607356

- Token level evaluation results on the test set were as follows:

  F1 = 0.9035459, Precision = 0.89318925, Recall = 0.91415083

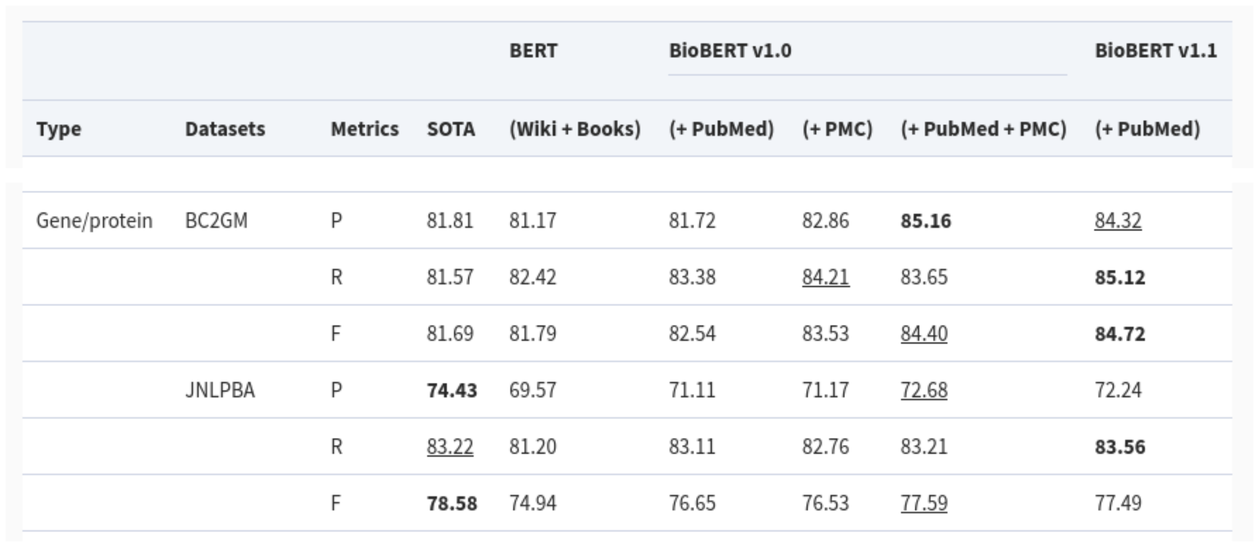
- Entity level evaluation results on the test set using BioBERT's evaluation script is as follows:



```
[33] !perl /content/drive/MyDrive/ner/biobert/biocodes/conlleval.pl < $OUTPUT_DIR/NER_result_conll.txt
```

```
processed 36334 tokens with 1748 phrases; found: 1792 phrases; correct: 1516.
accuracy:  97.82%; precision:  84.60%; recall:  86.73%; FB1:  85.65
             MISC: precision:  84.60%; recall:  86.73%; FB1:  85.65  1792
```

- BioBERT's test results in biomedical named entity recognition on GENE/protein data (Benchmarks) - using BioBERT's evaluation script is as follows:



				BERT	BioBERT v1.0			BioBERT v1.1
Type	**Datasets**	**Metrics**	**SOTA**	**(Wiki + Books)**	**(+ PubMed)**	**(+ PMC)**	**(+ PubMed + PMC)**	**(+ PubMed)**
Gene/protein	BC2GM	P	81.81	81.17	81.72	82.86	**85.16**	84.32
		R	81.57	82.42	83.38	84.21	83.65	**85.12**
		F	81.69	81.79	82.54	83.53	84.40	**84.72**
	JNLPBA	P	**74.43**	69.57	71.11	71.17	72.68	72.24
		R	83.22	81.20	83.11	82.76	83.21	**83.56**
		F	**78.58**	74.94	76.65	76.53	77.59	77.49

For complete results go to: https://academic.oup.com/view-large/199180401

Here, we can see that BioBERT fine-tuned on the provided dataset and then evaluated using BioBERT's entity level evaluation script shows slightly better F1 score performance than than the best benchmarks on BioBERT v1.1 for Gene/protein specific datasets with a score of 85.65.

- Entity level evaluation results on the test set using the evaluation script provided for this assignment is as follows:

  1748 entities in gold standard.
  1779 total entities found.
  1514 of which were correct.

Precision: 0.8510399100618324
Recall: 0.8661327231121282
F1-measure: 0.858519988658917

Hyperparameters

We fine-tuned our model for 10 epochs, with a batch size of 32 and a learning rate of 0.00005. The paper associated with the model suggests fine-tuning for approximately 50 epochs, but based on the merits of the results that we saw, we chose to terminate after 10 epochs, primarily due to a lack of compute resources. The model took ~ 40 minutes to fine-tune on a Tesla P100 and ~2 hours 10 mins on a Tesla K80.

Discussion

Based on evaluation of the test set on vanilla BioBERT followed by the saved weights at different fine-tuning steps, we see a gradual improvement in performance of the model with an increase in the number of fine-tuning steps.BioBERT shows signifcant improvement on vanilla BERT that we tried out previously in that it was able to correctly tag longer sequences.

We can explore if preserving the case of the text has an effect on the model performance or not. Also, given the availability of sufficient computing resources we would like to fine-tune this for 50 or more epochs as suggested by the authors and see if there is any improvement in performance (although overfitting might be more apparent in this case, given the small dataset).

Another avenue to explore could be using the pretrained BioBERT model on the large custom vocabulary and see if out of vocabulary words are better handled.

Final Submission

For the purpose of this assignment we will be submitting the system output from both Models 1 and 3. Our best performing model however is Model 3 which is finetuned on bioBERT.

- Model 3 Output File: bioBERT-output.txt
- Model 1 Output File: deep-crf-output.txt

Contribution

Nitin Kumar

- Nitin worked on Model 1 - DeepCRF - BiLSTM-CNN-CRF
- Debugged and validated results from Model 3
- Wrote pre and post processing scripts for the datasets and outputs
- Wrote one-third of this report.

Rohan Das

- Rohan worked on Model 3 - Fine-Tuning on BioBERT
- Performed hyperparameter tuning on Model 1
- Wrote pre and post processing scripts for the datasets and outputs
- Wrote one-third of this report.

Tushar Gautam

- Tushar worked on Model 2 - bert-base model
- Debugged and validated results from Model 3
- Wrote pre and post processing scripts for the datasets and outputs
- Wrote one-third of this report.

Bibliography

[1] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, *Bioinformatics*, Volume 36, Issue 4, 15 February 2020, Pages 1234–1240, https://doi.org/10.1093/bioinformatics/btz682

[2] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang, BioBERT, (2020), Github Repository, https://github.com/dmis-lab/biobert

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanov, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
Paper: https://arxiv.org/abs/1810.04805
Pre-trained Model: https://huggingface.co/bert-base-cased

[4] Xuezhe Ma and Eduard Hovy, End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF.
Paper: https://arxiv.org/pdf/1603.01354.pdf
Model Implementation with Extensions: https://github.com/aonotas/deep-crf